Preconditioning

Noisy, Ill-Conditioned Linear Systems

James G. Nagy
Emory University
Atlanta, GA

Outline

1. The Basic Problem
2. Regularization / Iterative Methods
3. Preconditioning
4. Example: Image Restoration
5. Summary
Basic Problem

Linear system of equations

\[ b = Ax \]

where

- \( A, b \) are known
- \( A \) is large, structured
- Goal: Compute an approximation of \( x \)
Basic Problem

Linear system of equations

\[ \mathbf{b} = A\mathbf{x} + \mathbf{e} \]

where

- \( A, \mathbf{b} \) are known
- \( A \) is large, structured, **ill-conditioned**
- Goal: Compute an approximation of \( \mathbf{x} \)
Basic Problem

Linear system of equations

\[ b = Ax + e \]

where

- \( A, b \) are known
- \( A \) is large, structured, \textit{ill-conditioned}
- Goal: Compute an approximation of \( x \)

Applications: Ill-posed inverse problems.

- Geomagnetic Prospecting
- Tomography
- Image Restoration
Basic Problem

Linear system of equations

\[ b = Ax + e \]

where

- \( A, b \) are known
- \( A \) is large, structured, ill-conditioned
- Goal: Compute an approximation of \( x \)

Applications: Ill-posed inverse problems.

- Geomagnetic Prospecting
- Tomography
- **Image Restoration**
  - \( b = \) observed image
  - \( A = \) blurring matrix (structured)
  - \( e = \) noise
  - \( x = \) true image
Basic Problem

Linear system of equations

\[ \mathbf{b} = A\mathbf{x} + \mathbf{e} \]

where

- \( A, \mathbf{b} \) are known
- \( A \) is large, structured, ill-conditioned
- Goal: Compute an approximation of \( \mathbf{x} \)

Applications: Ill-posed inverse problems.
- Geomagnetic Prospecting
- Tomography
- **Image Restoration**
  - \( \mathbf{b} = \) observed image
  - \( A = \) blurring matrix (structured)
  - \( \mathbf{e} = \) noise
  - \( \mathbf{x} = \) true image

Example →
Computational difficulties revealed through SVD:

Let \( A = U\Sigma V^T \) where

- \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) \), \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0 \)
- \( U^T U = I \), \( V^T V = I \)
Basic Problem – Properties

Computational difficulties revealed through SVD:

Let \( A = U\Sigma V^T \) where

- \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) \), \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0 \)
- \( U^TU = I \), \( V^TV = I \)

For ill-posed inverse problems,

- \( \sigma_1 \approx 1 \), small singular values cluster at 0
- small singular values \( \Rightarrow \) oscillating singular vectors
Inverse solution for noisy, ill-posed problems:

If $A = U\Sigma V^T$, then

$$x = A^{-1}b$$

$$= V\Sigma^{-1}U^Tb$$

$$= \sum_{i=1}^{n} \frac{u_i^Tb}{\sigma_i}v_i$$
Inverse solution for noisy, ill-posed problems:

If $A = U \Sigma V^T$, then

$$\hat{x} = A^{-1}(b + e)$$

$$= V \Sigma^{-1} U^T (b + e)$$

$$= \sum_{i=1}^{n} \frac{u_i^T (b + e)}{\sigma_i} v_i$$
Inverse solution for noisy, ill-posed problems:

If \( A = U \Sigma V^T \), then

\[
\hat{x} = A^{-1}(b + e)
\]

\[
= V \Sigma^{-1} U^T (b + e)
\]

\[
= \sum_{i=1}^{n} \frac{u_i^T (b + e)}{\sigma_i} v_i
\]

\[
= \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i + \sum_{i=1}^{n} \frac{u_i^T e}{\sigma_i} v_i
\]

\[
= x + \text{error}
\]
Inverse solution for noisy, ill-posed problems:

If \( A = U \Sigma V^T \), then

\[
\hat{x} = A^{-1}(b + e) \\
= V \Sigma^{-1} U^T (b + e) \\
= \sum_{i=1}^{n} \frac{u_i^T(b + e)}{\sigma_i} v_i \\
= \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i + \sum_{i=1}^{n} \frac{u_i^T e}{\sigma_i} v_i \\
= x + \text{error}
\]
Regularization

**Basic Idea:** Filter out effects of small singular values.

\[
x_{\text{reg}} = \sum_{i=1}^{n} \phi_i \frac{u_i^T b}{\sigma_i} v_i
\]

where the ”filter factors” satisfy

\[
\phi_i \approx \begin{cases} 
1 & \text{if } \sigma_i \text{ is large} \\
0 & \text{if } \sigma_i \text{ is small}
\end{cases}
\]
Some regularization methods:

1. Truncated SVD

\[ x_{\text{tsvd}} = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i \]

2. Tikhonov

\[ x_{\text{tik}} = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \frac{u_i^T b}{\sigma_i} v_i \]

3. Wiener

\[ x_{\text{wien}} = \sum_{i=1}^{n} \frac{\delta_i \sigma_i^2}{\delta_i \sigma_i^2 + \gamma_i^2} \frac{u_i^T b}{\sigma_i} v_i \]
Iterative Regularization

Basic idea:

• Use an iterative method (e.g., conjugate gradients)

• Terminate iteration before theoretical convergence:
  – Early iterations reconstruct solution.
  – Later iterations reconstruct noise.
Iterative Regularization

Basic idea:
• Use an iterative method (e.g., conjugate gradients)

• Terminate iteration before theoretical convergence:
  – Early iterations reconstruct solution.
  – Later iterations reconstruct noise.

Some important methods:
• CGLS, LSQR, GMRES

• MR2 (Hanke, ’95)

• MRNSD (Kaufman, ’93; N., Strakos, ’00)
Iterative Regularization

Efficient for large problems, provided

1. Multiplication with $A$ is not expensive.

2. Convergence is rapid enough.
Iterative Regularization

Efficient for large problems, provided

1. Multiplication with $A$ is not expensive.
   
   Image restoration $\Leftrightarrow$ Use FFTs

2. Convergence is rapid enough.
   
   - CGLS, LSQR, GMRES, MR2 often fast, especially for severely ill-posed, noisy problems.
   - MRNSD based on steepest descent, typically converges very slowly.
Iterative Regularization

Efficient for large problems, provided

1. Multiplication with $A$ is not expensive.

   Image restoration $\iff$ Use FFTs

2. Convergence is rapid enough.

   - CGLS, LSQR, GMRES, MR2 often fast, especially for severely ill-posed, noisy problems.

   - MRNSD based on steepest descent, typically converges very slowly.

Example →
Preconditioning

Purposes of preconditioning:

1. Accelerate convergence.
   
   • Apply iterative method to $P^{-1}Ax = P^{-1}b$. 

Preconditioning

Purposes of preconditioning:

1. Accelerate convergence.
   - Apply iterative method to $P^{-1}Ax = P^{-1}b$.
   - In this case we minimize $\|x\|_2$.

2. Enforce regularization constraint on solution.
   (Hanke, '92; Hansen, '98)
   - Apply iterative method to $AL^{-1}Lx = b$.
   - In this case, we minimize $\|Lx\|_2$. 
Preconditioning for Regularization

Basic idea:

- Find a matrix $L$ to enforce smoothness constraint

$$\min ||Lx||_2$$

- Typically $L$ approximates a derivative operator.
Typical approach for $Ax = b$

- Find matrix $P$ so that $P^{-1}A \approx I$.

- "Ideal" choice: $P = A$
  
  In this case, converge in one iteration to $x = A^{-1}b$
Typical approach for $Ax = b$

- Find matrix $P$ so that $P^{-1}A \approx I$.

- "Ideal" choice: $P = A$
  
  In this case, converge in one iteration to $x = A^{-1}b$

For ill-conditioned, noisy problems

- Inverse solution is corrupted with noise
Preconditioning for Speed

Typical approach for $Ax = b$

- Find matrix $P$ so that $P^{-1}A \approx I$.

- "Ideal" choice: $P = A$
  In this case, converge in one iteration to $x = A^{-1}b$

For ill-conditioned, noisy problems

- Inverse solution is corrupted with noise

- "Ideal" regularized preconditioner: If $A = U\Sigma V^T$
  (Hanke, N., Plemmons, ’93)

  $$P = U\Sigma_k V^T = U \text{diag}(\sigma_1, \ldots, \sigma_k, 1, \ldots, 1)V^T$$
Notice that the preconditioned system is:

\[ P^{-1}A = (U\Sigma_k V^T)^{-1}(U\Sigma V^T) \]

\[ = V\Sigma_k^{-1}\Sigma V^T \]

\[ = V\Delta V^T \]

where \( \Delta = \text{diag}(1, \ldots, 1, \sigma_{k+1}, \ldots, \sigma_n) \)

That is,

- Large (good) singular values clustered at 1.
- Small (bad) singular values not clustered.
Remaining questions:

1. How to choose truncation index, $k$?

   Use regularization parameter choice methods, e.g., GCV, L-curve, Picard condition

2. We can’t compute SVD, so now what?

   Use SVD approximation.
Preconditioning for Speed

An SVD Approximation:

- Decompose $A$ as: (Van Loan and Pitsianis, ’93)

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

where $C_1 \otimes D_1 = \text{argmin} \|A - C \otimes D\|_F$. 
An SVD Approximation:

- Decompose $A$ as: (Van Loan and Pitsianis, ’93)
  \[ A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k \]
  where \( C_1 \otimes D_1 = \arg\min ||A - C \otimes D||_F \).

- Choose a “structured” (or sparse) $U$ and $V$. 

An SVD Approximation:

- Decompose $A$ as: (Van Loan and Pitsianis, ’93)
  \[ A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k \]
  where \[ C_1 \otimes D_1 = \operatorname{argmin}_{C} \|A - C \otimes D\|_F. \]

- Choose a “structured” (or sparse) $U$ and $V$.

- Let $\Sigma = \operatorname{argmin}_{\Sigma} \|A - U\Sigma V^T\|_F.$
An SVD Approximation:

- Decompose $A$ as: (Van Loan and Pitsianis, ’93)
  \[ A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k \]
  where $C_1 \otimes D_1 = \arg\min \|A - C \otimes D\|_F$.

- Choose a "structured" (or sparse) $U$ and $V$.

- Let $\Sigma = \arg\min \|A - U\Sigma V^T\|_F$.

  That is,
  \[ \Sigma = \text{diag}(U^TAV) \]
Preconditioning for Speed

An SVD Approximation:

- Decompose $A$ as: (Van Loan and Pitsianis, ’93)
  $$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$
  where $C_1 \otimes D_1 = \arg\min ||A - C \otimes D||_F$.

- Choose a “structured” (or sparse) $U$ and $V$.

- Let $\Sigma = \arg\min ||A - U\Sigma V^T||_F$.

That is,

$$\Sigma = \text{diag} \left( U^T A V \right)$$

$$= \text{diag} \left( U^T \left( \sum_{i=1}^{k} C_i \otimes D_i \right) V \right)$$
Preconditioning for Speed

Choices for $U$ and $V$ depend on problem (application).

- Since

$$A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k$$

and

$$C_1 \otimes D_1 = \text{argmin} \|A - C \otimes D\|_F$$

we might use singular vectors of $C_1 \otimes D_1$. 
Choices for $U$ and $V$ depend on problem (application).

- Since

\[ A = C_1 \otimes D_1 + C_2 \otimes D_2 + \cdots + C_k \otimes D_k \]

and

\[ C_1 \otimes D_1 = \arg\min ||A - C \otimes D||_F \]

we might use singular vectors of $C_1 \otimes D_1$.

- For image restoration, we also use

  Fourier Transforms (FFTs)

  Discrete Cosine Transforms (DCTs)
Example: Image Restoration

1. Matrix Structure

2. Efficiently computing SVD approximation
Matrix Structure in Image Restoration

First, how do we get the matrix, $A$?
First, how do we get the matrix, $A$?

- Using linear algebra notation, the $i$-th column of $A$ can be written as:

$$Ae_i = \begin{bmatrix} a_1 & \cdots & a_i & \cdots & a_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = a_i$$
Matrix Structure in Image Restoration

First, how do we get the matrix, $A$?

- Using linear algebra notation, the $i$-th column of $A$ can be written as:

$$Ae_i = \begin{bmatrix} a_1 & \cdots & a_i & \cdots & a_n \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = a_i$$

- In an imaging system,

$$e_i = \text{point source}$$
$$Ae_i = \text{point spread function (PSF)}$$
Matrix Structure in Image Restoration

point source

PSF
Matrix Structure in Image Restoration

Spatially invariant PSF implies:

\[ e_i \quad e_j \quad e_k \]

\[ Ae_i \quad Ae_j \quad Ae_k \]
That is, spatially invariant implies

- Each column of $A$ is identical, modulo shift.
- One point PSF is enough to fully describe $A$.
- $A$ has Toeplitz structure.
Matrix Structure in Image Restoration

\[
e_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \text{blur} \rightarrow \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow Ae_5 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}
\]

\[
A = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}
\]
Matrix Structure in Image Restoration

\[ e_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \text{blur} \rightarrow \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow A e_5 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \]
Matrix Structure in Image Restoration

\[ e_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \text{blur} \rightarrow \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow A e_5 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \]

\[ A = \begin{bmatrix} p_{22} & p_{21} & p_{22} & p_{12} & p_{11} \\ p_{23} & p_{22} & p_{21} & p_{13} & p_{12} \\ p_{23} & p_{22} & p_{21} & p_{13} & p_{12} \\ p_{32} & p_{31} & p_{32} & p_{22} & p_{12} \\ p_{33} & p_{32} & p_{31} & p_{23} & p_{12} \\ p_{33} & p_{32} & p_{31} & p_{23} & p_{12} \\ p_{32} & p_{31} & p_{32} & p_{22} & p_{12} \\ p_{33} & p_{32} & p_{31} & p_{23} & p_{12} \end{bmatrix} \]
Matrix Structure in Image Restoration

Matrix Summary

<table>
<thead>
<tr>
<th>Boundary Condition</th>
<th>Matrix Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>BTTB</td>
</tr>
<tr>
<td>periodic</td>
<td>BCCB</td>
</tr>
<tr>
<td>reflexive</td>
<td>BTTB+BTCB+BHTB+BHHB</td>
</tr>
</tbody>
</table>

B = block  
T = Toeplitz  
C = circulant  
H = Hankel
## Matrix Summary

<table>
<thead>
<tr>
<th>Boundary Condition</th>
<th>Matrix Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>BTTB</td>
</tr>
<tr>
<td>periodic</td>
<td>BCCB (use FFT)</td>
</tr>
<tr>
<td>reflexive</td>
<td>BTTB + BTHB + BHTB + BHHB (use DCT)</td>
</tr>
</tbody>
</table>

**Abbreviations:**

- \( B \) = block
- \( T \) = Toeplitz
- \( C \) = circulant
- \( H \) = Hankel
Matrix Structure in Image Restoration

For a separable PSF, we get:

\[
\begin{bmatrix}
p_{11} & p_{12} & p_{13} 
p_{21} & p_{22} & p_{23} 
p_{31} & p_{32} & p_{33}
\end{bmatrix} = \mathbf{c} \mathbf{d}^T = 
\begin{bmatrix}
c_1 d_1 & c_1 d_2 & c_1 d_3 
c_2 d_1 & c_2 d_2 & c_2 d_3 
c_3 d_1 & c_3 d_2 & c_3 d_3
\end{bmatrix} \rightarrow A e_5 = 
\begin{bmatrix}
c_1 
\begin{pmatrix}
d_1 
d_2 
d_3
\end{pmatrix}
c_2 
\begin{pmatrix}
d_1 
d_2 
d_3
\end{pmatrix}
c_3 
\begin{pmatrix}
d_1 
d_2 
d_3
\end{pmatrix}
\end{bmatrix}
\]
For a separable PSF, we get:

\[
\begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & p_{22} & p_{23} \\
p_{31} & p_{32} & p_{33}
\end{bmatrix} = \mathbf{cd}^T = \begin{bmatrix}
c_1 d_1 & c_1 d_2 & c_1 d_3 \\
c_2 d_1 & c_2 d_2 & c_2 d_3 \\
c_3 d_1 & c_3 d_2 & c_3 d_3
\end{bmatrix} \rightarrow \mathbf{A} e_5 = \begin{bmatrix}
c_1 & \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\
c_2 & \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \\
c_3 & \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}
\end{bmatrix}
\]
For a separable PSF, we get:

\[
\begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & p_{22} & p_{23} \\
p_{31} & p_{32} & p_{33}
\end{bmatrix} = cd^T =
\begin{bmatrix}
c_1d_1 & c_1d_2 & c_1d_3 \\
c_2d_1 & c_2d_2 & c_2d_3 \\
c_3d_1 & c_3d_2 & c_3d_3
\end{bmatrix} \rightarrow Ae_5 =
\begin{bmatrix}
c_1 \\
c_2 \\
c_3
\end{bmatrix}
\begin{pmatrix}
d_1 \\
d_2 \\
d_3
\end{pmatrix}
\]
Matrix Structure in Image Restoration

For a separable PSF, we get:

\[
\begin{bmatrix}
  p_{11} & p_{12} & p_{13} \\
  p_{21} & p_{22} & p_{23} \\
  p_{31} & p_{32} & p_{33}
\end{bmatrix} = \mathbf{c}\mathbf{d}^T = \begin{bmatrix}
  c_1d_1 & c_1d_2 & c_1d_3 \\
  c_2d_1 & c_2d_2 & c_2d_3 \\
  c_3d_1 & c_3d_2 & c_3d_3
\end{bmatrix} \rightarrow \mathbf{Ae}_5 = \begin{bmatrix}
  c_1 \begin{pmatrix}
    d_1 \\
    d_2 \\
    d_3
  \end{pmatrix} \\
  c_2 \begin{pmatrix}
    d_1 \\
    d_2 \\
    d_3
  \end{pmatrix} \\
  c_3 \begin{pmatrix}
    d_1 \\
    d_2 \\
    d_3
  \end{pmatrix}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  c_2 \begin{pmatrix}
    d_2 & d_1 & d_3 \\
    d_3 & d_2 & d_1 \\
    d_1 & d_3 & d_2
  \end{pmatrix} \\
  c_3 \begin{pmatrix}
    d_2 & d_1 & d_3 \\
    d_3 & d_2 & d_1 \\
    d_1 & d_3 & d_2
  \end{pmatrix}
\end{bmatrix} \begin{bmatrix}
  c_1 \begin{pmatrix}
    d_2 & d_1 & d_3 \\
    d_3 & d_2 & d_1 \\
    d_1 & d_3 & d_2
  \end{pmatrix} \\
  c_2 \begin{pmatrix}
    d_2 & d_1 & d_3 \\
    d_3 & d_2 & d_1 \\
    d_1 & d_3 & d_2
  \end{pmatrix} \\
  c_3 \begin{pmatrix}
    d_2 & d_1 & d_3 \\
    d_3 & d_2 & d_1 \\
    d_1 & d_3 & d_2
  \end{pmatrix}
\end{bmatrix} = \mathbf{C} \otimes \mathbf{D}
\]
If the PSF is not separable, we can still compute:

\[ P = \sum_{i=1}^{r} c_i d_i^T \]  

(sum of rank-1 matrices)

and therefore, get

\[ A = \sum_{i=1}^{r} C_i \otimes D_i \]  

(sum of Kron. products)

In fact, we can get “optimal” decompositions.

(Kamm, N, ’00; N., Ng, Perrone, 03)
SVD Approximation for Image Restoration

Use $A \approx U \Sigma V^T$, where

- If $F \left( \sum C_i \otimes D_i \right) F^*$ is best,
  
  $$U = V = F^*, \quad \Sigma = \text{diag} \left( F \left( \sum C_i \otimes D_i \right) F^* \right)$$

- If $C \left( \sum C_i \otimes D_i \right) C^T$ is best,
  
  $$U = V = C^T, \quad \Sigma = \text{diag} \left( C \left( \sum C_i \otimes D_i \right) C^T \right)$$

- If $(U_c \otimes U_d)^T \left( \sum C_i \otimes D_i \right) (V_c \otimes V_d)$ is best,
  
  $$U = U_c \otimes U_d, \quad V = V_c \otimes V_d,$$
  
  $$\Sigma = \text{diag} \left( (U_c \otimes U_d)^T \left( \sum C_i \otimes D_i \right) (V_c \otimes V_d) \right)$$

Example →
• Preconditioning ill-posed problems is difficult, but possible.

• Can build approximate SVD from Kronecker product approximations.

• Can implement efficiently for image restoration.

• Matlab software: RestoreTools (Lee, N., Perrone, ’02)

  Object oriented approach for image restoration.
  http://www.mathcs.emory.edu/~nagy/RestoreTools/

  Related software for ill-posed problems
  (Hansen, Jacobsen)
  http://www.imm.dtu.dk/~pch/Regutools/